Chapitre 0 : la boîte à outils pour commencer à "pythoner" (0c13-1780985)

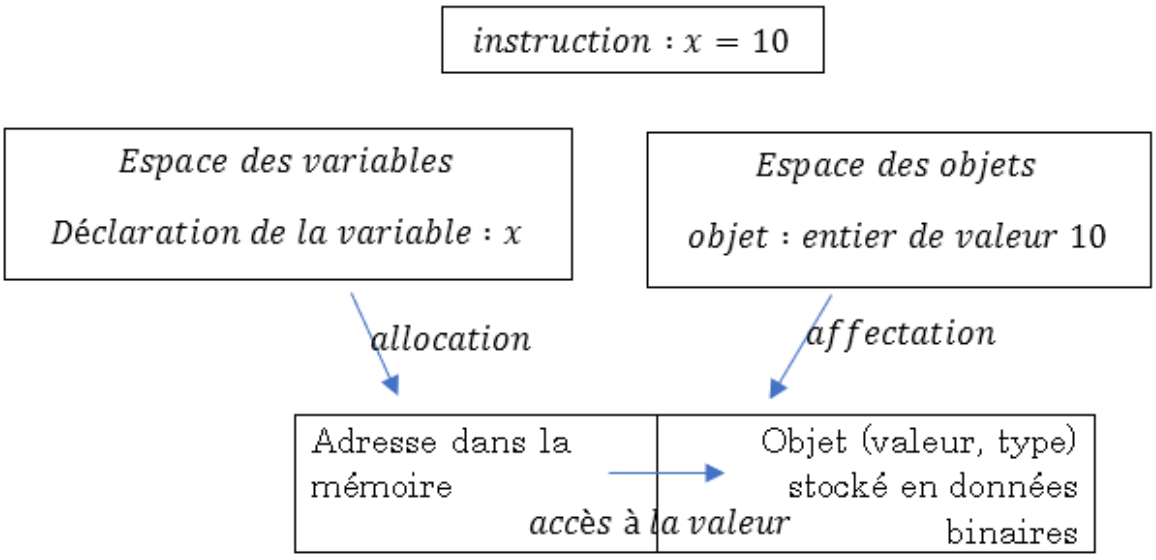
Quelques généralités :

- Pour installer python sous windows : http://winpython.github.io/ (http://winpython.github.io/)
- pour installer python sous mac ou lynux : https://pyzo.org/start.html (https://pyzo.org/start.html)
- plusieurs IDE sont possibles : spyder, pyzo, pycharm, jupyther, tonny,...
- Vous pouvez aussi travailler en ligne avec python tutor
- Vous pouvez travailler sur l'ENT avec Capytal

I- Opérateur d'affectation "=" (code quizinière : DMK4L7)

Analysons les effets de l'instruction très simple $x = 10$:

- 1) Parmi tous les **objets** possibles (entiers, nombre décimaux, données textuelles, ...), c'est l'objet "entier 10" qui a été choisi (avec toutes ses propriétés mathématiques)
- 2) Une variable x est **déclarée**
- 3) Le signe "=" réalise **une allocation** (un espace réservé dans la mémoire) et **une affectation** (x est liée à la valeur 10 par l'adresse mémoire) : on dit que la variable x référence l'objet 10 dans la mémoire



In [1]:
```
# Exemple
x=10

#
```

In [2]:
```
y=x

#
```

In [3]:
```
x=11.0



#
```

In [16]:
```
z=x+10
print(z)#on a accès à la valeur de x pour d'autres calculs
```

```
21.0
```

In [4]:
```
x=x+10


```

II- Les opérations numériques sur les entiers et les nombres décimaux (code quizinière : 99BZ9Y)

In [2]:
```
x,y=10,3
#la somme
print(x+y)
```

```
13
```

In [3]:
```
#la soustraction
print(x-y)
```

```
7
```

In [4]:
```
#produit de x par y
print(x*y)
```

```
30
```

In [5]:
```
# x puissance y


#
```

Rappels sur la division enclidienne :

- le dividende a : quanitité à diviser
- le diviseur b : quantité par laquelle on divise
- le quotient Q, le reste R : a//b, a%b tels que a=b*Q+R

In [6]:
```
# Division de x par y

```

In [7]:
```
# Quotient Q de la division entière de x par y

```

In [8]:
```
# Reste de la division de x par y

```

In [9]:
```
# doublet (Quotient, Reste)

```

In [10]:
```
#valeur absolue

```

In [11]:
```
#Conversion d'un nombre décimal en entier le plus proche de 0


```

In [12]:
```
#conversion d'un entier en nombre décimal

```

III- Les booléens (code quizinière : E7Y8ZQ)

- x or y : Vrai si x et/ou y sont vrais

In [33]:
```
print(True or True)#True
print(False or False)# False
print(False or True)#True
print(True or False)#True
print(True or rien)#True car opérateur paresseux (pas d'évaluation
```

```
True
False
True
True
True
```

- x and y : Vrai si x et y sont vrais

In [9]:
```
print(True and False)#False
print(False and False)# False
print(False and True)#False
print(False and rien)#False car opérateur paresseux (pas d'évaluati
print(True and True)# True
```

```
False
False
False
False
True
```

- not x : Vrai si x faux, faux si x vrai

In [13]:
```
print(not False)#True
print(not True)#False
#les booléens peuvent être vis comme un sous type des entiers
print(1==True)#True
print(0==False)#True
print(not 0)#True
print(not 1)#False
```

```
True
False
True
True
True
False
```

A noter que ces opérateurs suivent un ordre de priorité (not>and>or)

Certains opérateurs sur les types numériques renvoient un résultat booléen :

In [1]:
```
x=10
y=3
print(x<y)# x est-il plus strictement plus petit que y ?
print(x>y)# x est-il plus strictement plus grand que y ?
print(x<=y)# x est-il inférieur ou égale à y ?
print(x>=y)# x est-il inférieur ou égale à y ?
print(x==y)# x est-il identique à y ?
print (x!=y)# x est-il différent de y ?
```

```
False
True
False
True
False
True
```

IV- Les fonctions (code quizinière : OXD95W)

La définition d'une fonction se fait à l'aide du mot clé def :

In [14]:
```
#
#
```

In [15]:
```
#exemple f1(x)=2x+4

#Pour généraliser à toute fonction affine
```

In [16]:
```
#Rq : renvoyer et afficher un résultat :
# sous jupyter : si la cellule ne contient qu'une fonction alors le
# =>le print n'est pas n'est pas nécessaire en présence du return
```

On distingue les variables locales et globales :

- les variables globales ne sont pas définies dans le corps de la fonction. S'il s'agit d'un nombre, cette variable ne peut être modifiée,
- les variables locales sont définies dans le corps de la fonction et leur lecture n'est possible que pendant l'appel de cette fonction

Si une variable est appelées dans le corps de la fonction alors elle est d'abord recherchées dans l'espace des varaibles locales puis globales

In [17]:
```
#exemple 1 : si a n'est que variable globale

#
```

In [18]:
```
#exemple 2 : si a est variable locale et globale

#
```

In [19]:
```
#exemple 3 : si a est passé en argument alors il appartient à l'esp
#
```

V- Les chaînes de caractères

Les données textuelles ou chaînes de caractères ont les propriétés suivantes :

- création :

In [20]:
```
#
#
```

- Conversion:

In [21]:
```
#Effectué par le mot clé $str$
#
```

- Sélection ou tranche

In [22]:
```
#
#
```

- longueur d'une chaîne

In [23]:
```
#
```

VI- Structures conditionnelles (code quizinière : 54AXL3)

On a typiquement le squelette suivant :

In [25]:
```
x=float(input("rentre un nombre :"))
if x >0 :#on notera la présence d'une indentation
    print ("x est strictement positif")#instruction réalisée si la
elif x<0 :
    print ("x est strictement négatif")#instruction réalisée si la
else :
    print("x est nul")#instruction réalisée si les deux autres cond
```

```
rentre un nombre : 10

x est strictement positif
```

Rq : if et elif sont bien différents : un if est toujours testé, un elif et un else ne sont pas testés si le if précédent est True

In [26]:
```
x=11
if x >0 :#présence d'une indentation
    print ("x est strictement positif")#instruction réalisée si la
if x<10:
    print ("x est plus petit que 10")#instruction réalisée si la 2e
elif x<5 :
    print ("x est plus petit que 5")#instruction réalisée si la 2e
else :
    print("x est nul")#instruction réalisée si les deux autres cond
```

```
x est strictement positif
x est nul
```